浙江大学

本科实验报告

课程名称：计算机组成
姓名：曾一欣

学院：竺可桢学院

班级：求是科学班（计算机科学与技术）

专业：计算机科学与技术

学号：3180105144

指导老师：姜晓红
2020.7.9

浙江大学实验报告

课程名称： 计算机组成　　　实验类型：综合

实验项目名称：bonus mul/div/中断

学生姓名：曾一欣　　专业：计算机科学与技术　　学号： 3180105144

同组学生姓名： None　　　指导老师： 姜晓红

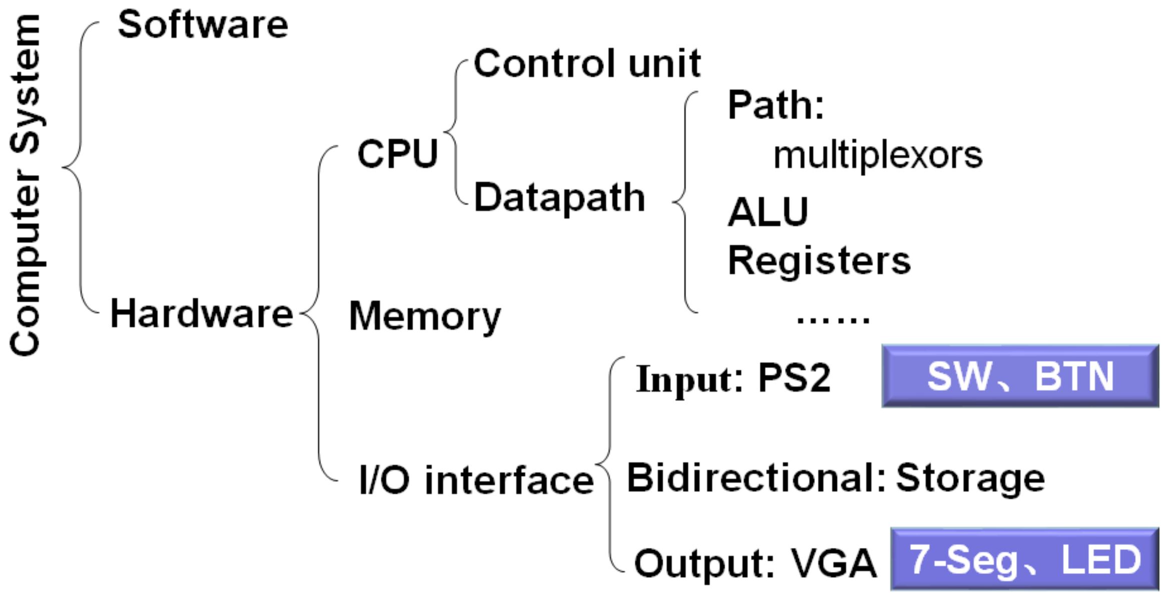
实验地点：无　　实验日期： 2020 年 7 月 9 日

一、 实验目的和要求

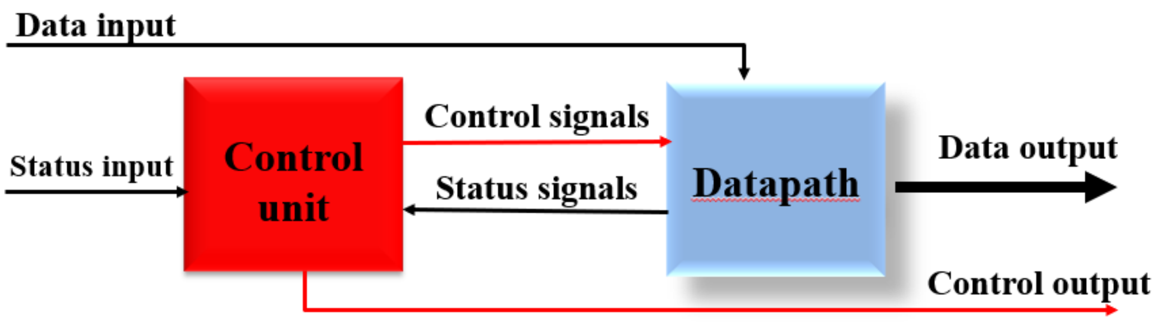
1. 设计乘法器和除法器，设计mul和div语句执行的datapath和ctrl
2. 设计简单的中断方式，实现eret指令

二、 实验内容和原理

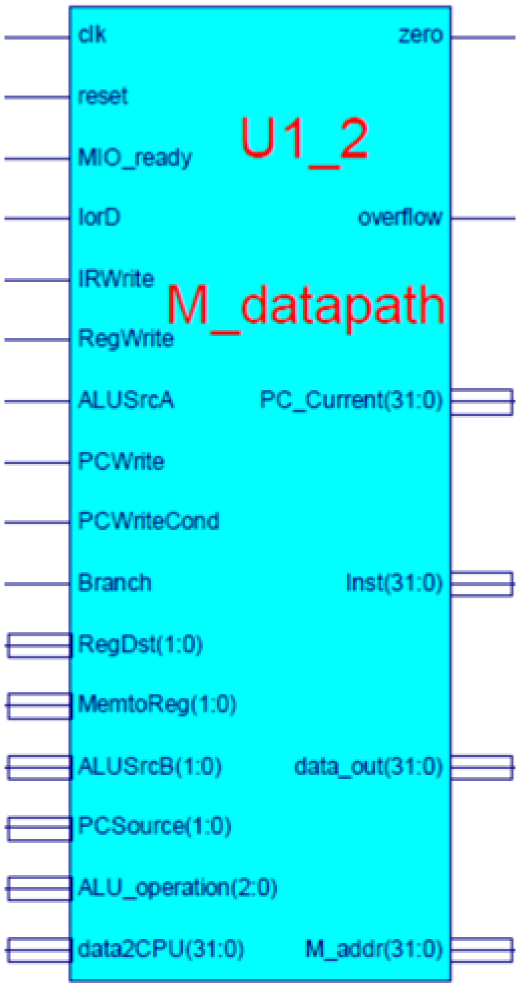
Decomposability of computer systems



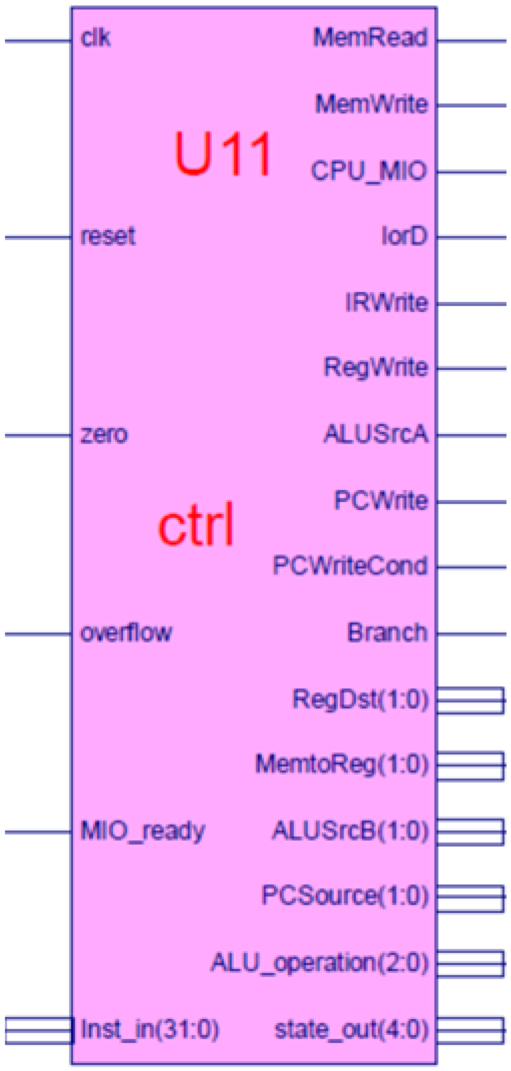
Digital circuit



DataPath



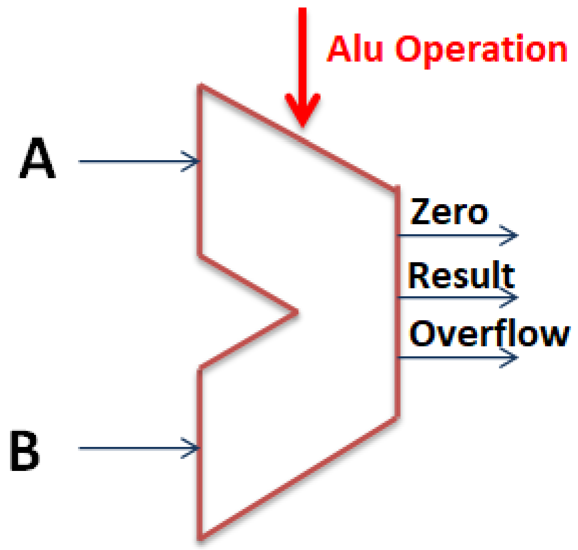
Controller



ALU

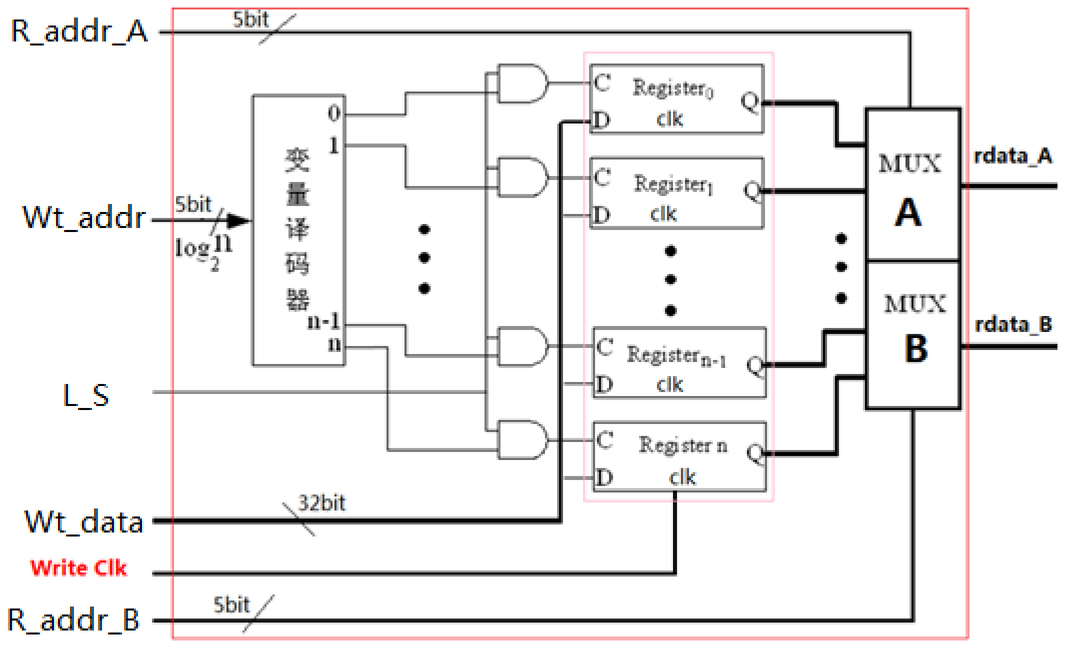
use the same module ALU in SCPU

ALU Control Lines	Function	note
000	And	兼容
001	Or	兼容
010	Add	兼容
110	Sub	兼容
111	Set on less than	
100	nor	扩展
101	srl	扩展
011	xor	扩展

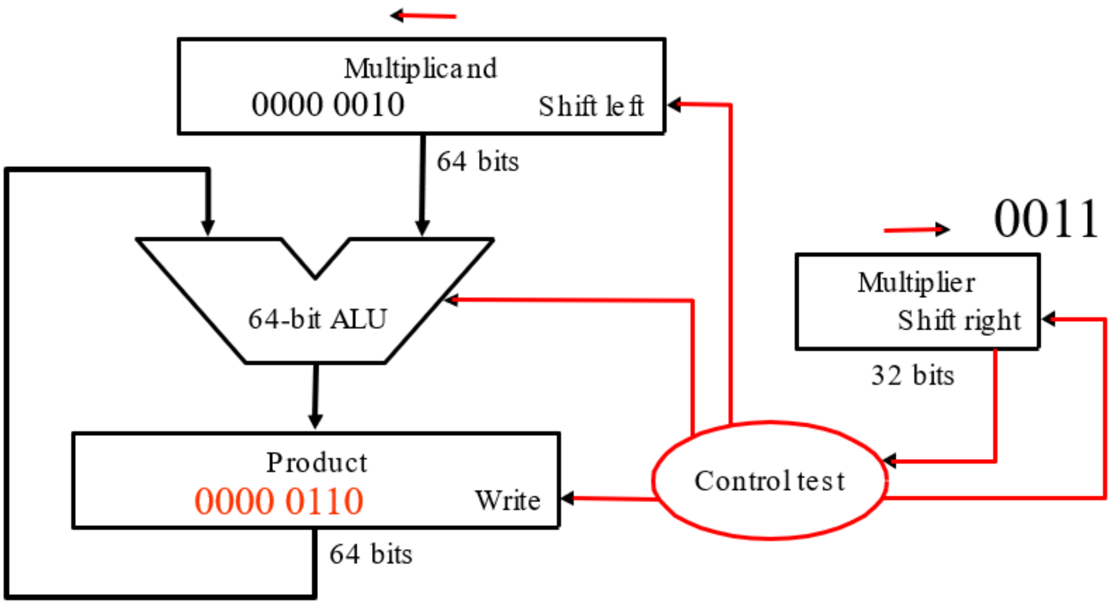


Register file

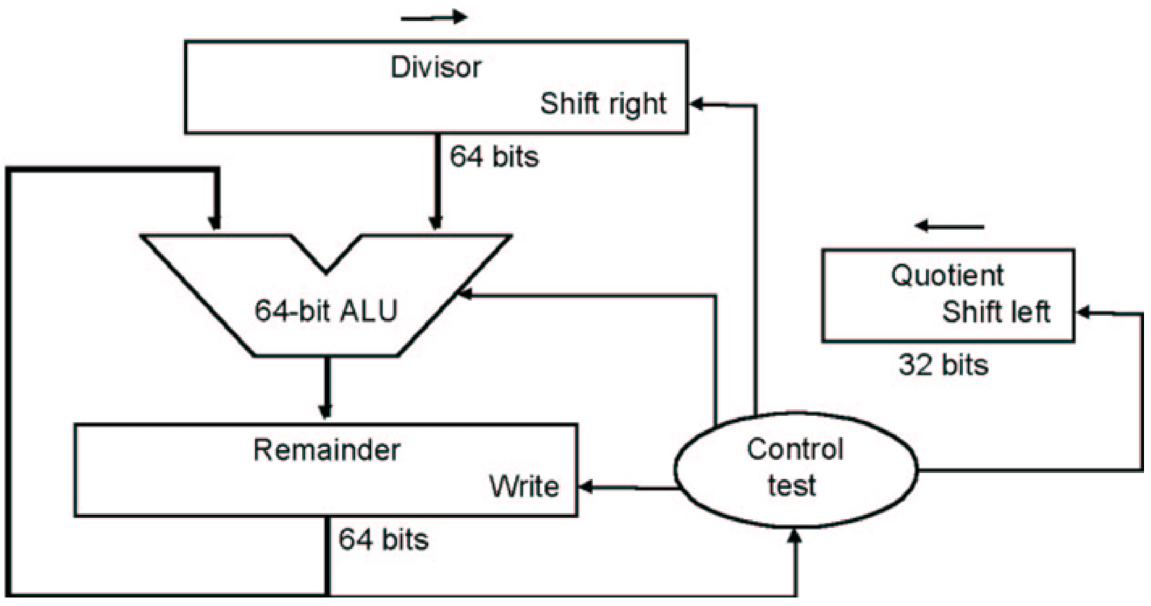
use the same module in SCPU



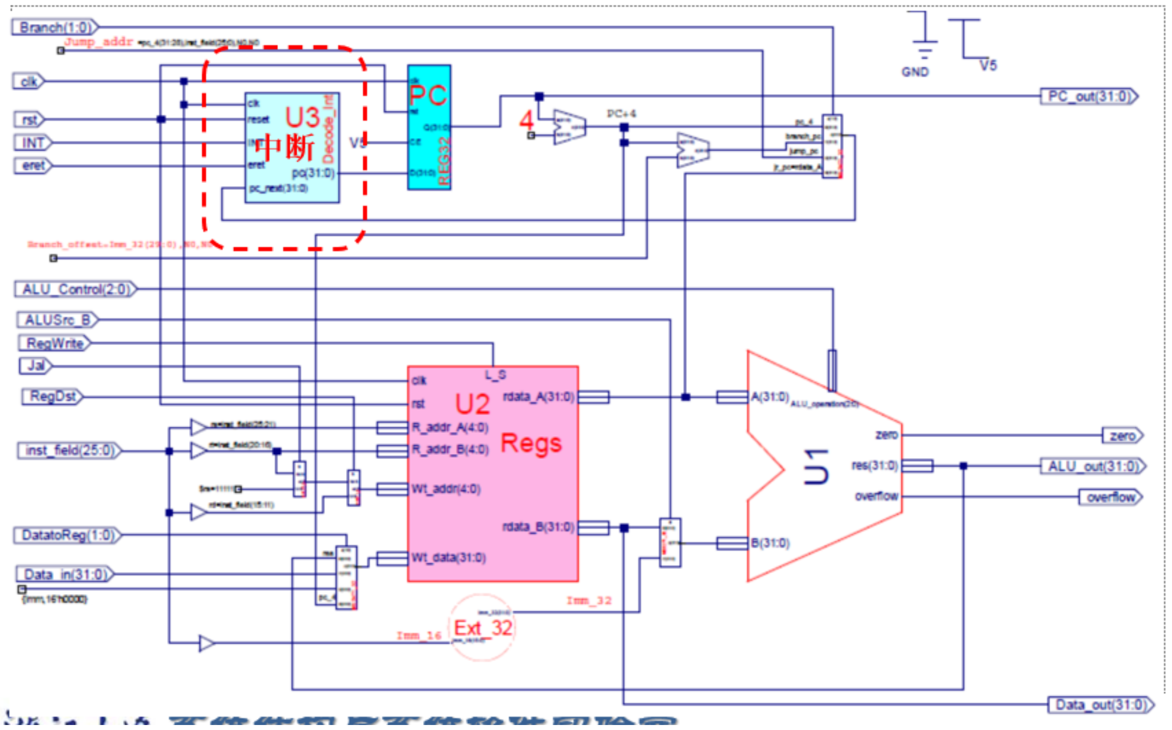
mul

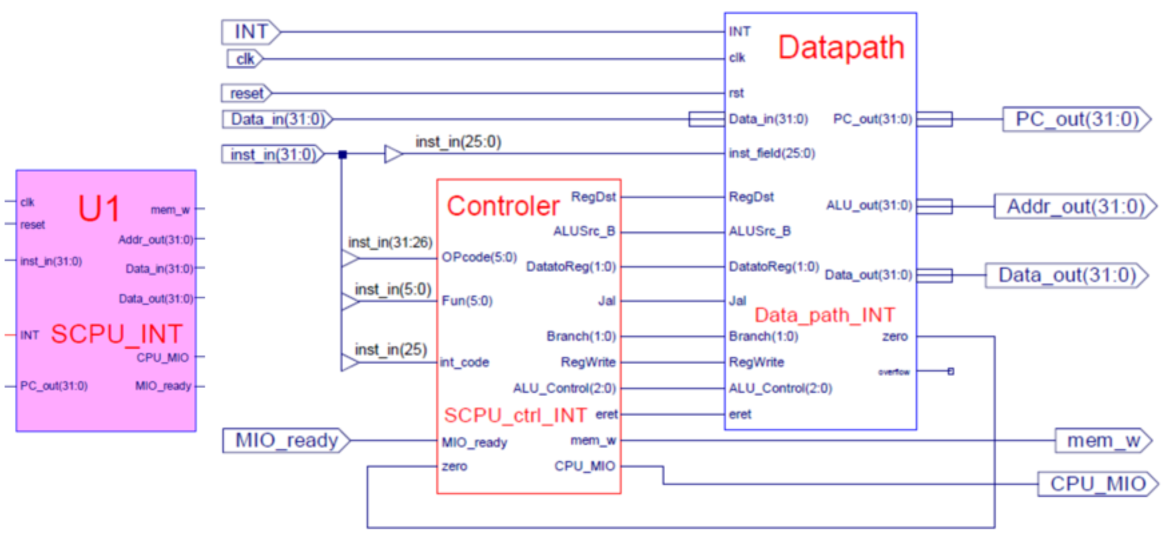


div


Divisor
Shift right
64 bits
64-bit ALU
Quotient
Shift left
32 bits
Remainder
Write
Control
test
64 bits

中断


Branch(1:0)
Jump_addr
clk
rst
INT
eret
U3
中断
Decode_Int
PC
REG32
4
PC+4
GND
V5
PC_out(31:0)
ALU_Control(2:0)
ALUSrc_B
RegWrite
Jal
RegDst
inst_field(25:0)
DatatoReg(1:0)
Data_in(31:0)
U2
Regs
L_S
rdata_A(31:0)
rdata_B(31:0)
R_addr_A(4:0)
R_addr_B(4:0)
Wt_addr(4:0)
Wt_data(31:0)
U1
A(31:0)
B(31:0)
zero
res(31:0)
overflow
ALU_out(31:0)
Data_out(31:0)
Ext_32
Imm_16
Imm_32

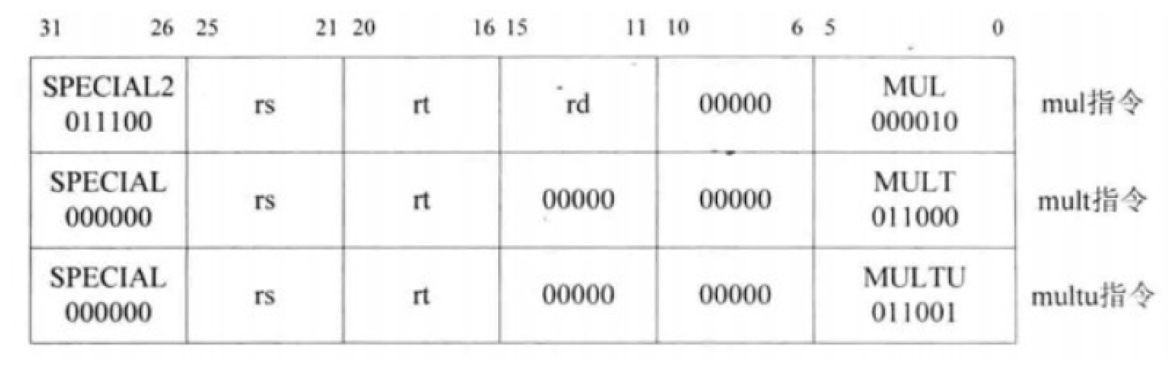


三、 实验过程和数据记录及结果分析

Search for the structure of mul, div and eret

mul

multu、mult，mul 指令



31 26	25 21	20 16	15 11	10 6	5 0	
SPECIAL2 011100	rs	rt	rd	00000	MUL 000010	mul指令
SPECIAL 000000	rs	rt	00000	00000	MULT 011000	mult指令
SPECIAL 000000	rs	rt	00000	00000	MULTU 011001	multu指令

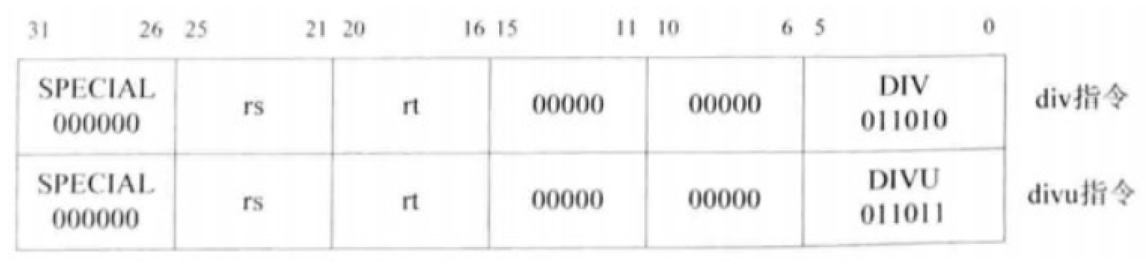
- 当指令码为SPECIAL2，功能码为6b000010时，表示mul指令，乘法运算。

指令用法为：mul rd，rs，st。

指令作用为：rd ← rs×rt，将地址为rs的通用寄存器的值与地址为rt的通用寄存器的值作为有符号数相乘，乘法结果的低32bit保存到地址为rd的通用寄存器中。

div

div、divu指令



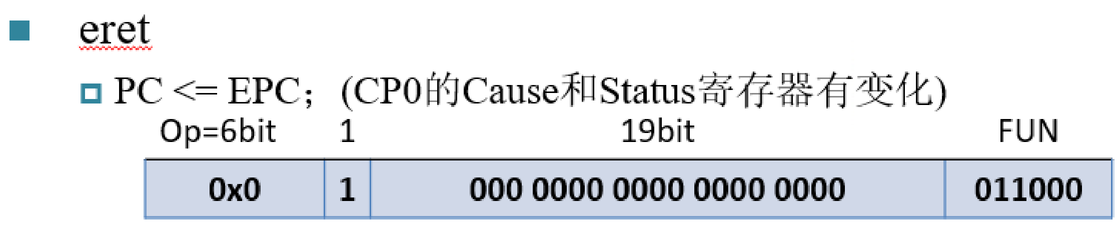
31 26	25 21	20 16	15 11	10 6	5 0	
SPECIAL 000000	rs	rt	00000	00000	DIV 011010	div指令
SPECIAL 000000	rs	rt	00000	00000	DIVU 011011	divu指令

- 当功能码是6b011010时，表示是div指令，有符号除法运算。

指令用法为：div rs，rt。

指令作用为：{HI，LO} ← rs/rt，将地址为rs的通用寄存器的值，与地址为rt的通用寄存器的值，作为有符号数进行除法运算，将商保存到寄存器LO，余数保存到寄存器HI。

eret



- eret
  - PC <= EPC；（CP0的Cause和Status寄存器有变化）

Op=6bit	1	19bit	FUN
0x0	1	000 0000 0000 0000 0000	011000

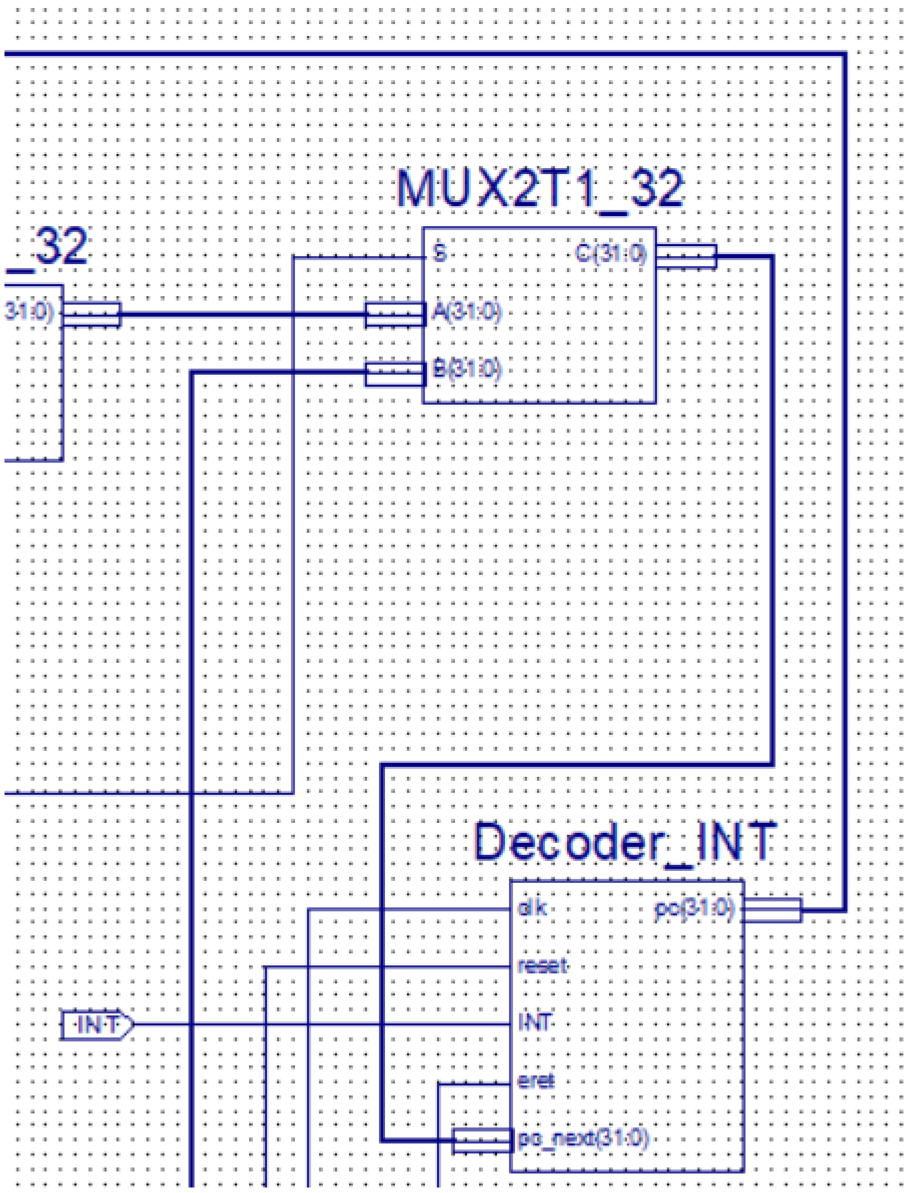
Modules

decoder INT

```
module Decoder_INT(
        input clk,
        input reset,
        input INT,
      input eret,
        input [31:0] pc_next,
        output reg [31:0] pc
        );

reg doing = 0, cando = 1, newint = 0;
reg [31:0] EPC;

always @(posedge clk or posedge reset)
        begin
         if (reset)
                begin
                    doing <= 0;
                    cando <= 1;
                    EPC <= 0;
                end
            else if (cando & newint)
                begin
                    doing <= 1;
                    cando <= 0;
                    EPC <= pc_next;
                end
            else begin
                doing <= 0;
                if (eret) cando <= 1;
            end
        end

wire clr;
assign clr = reset | doing;
always @(posedge INT or posedge clr)
    begin
        if (clr) newint <= 0;
        else newint <= 1;
    end
always @*
    begin
        if (reset)
            pc <= 32'h00000000;
        else if (cando & newint) pc <= 32'h00000004;
        else if (eret) pc <= EPC;
      else pc <= pc_next;
    end
endmodule
```

change of datapath

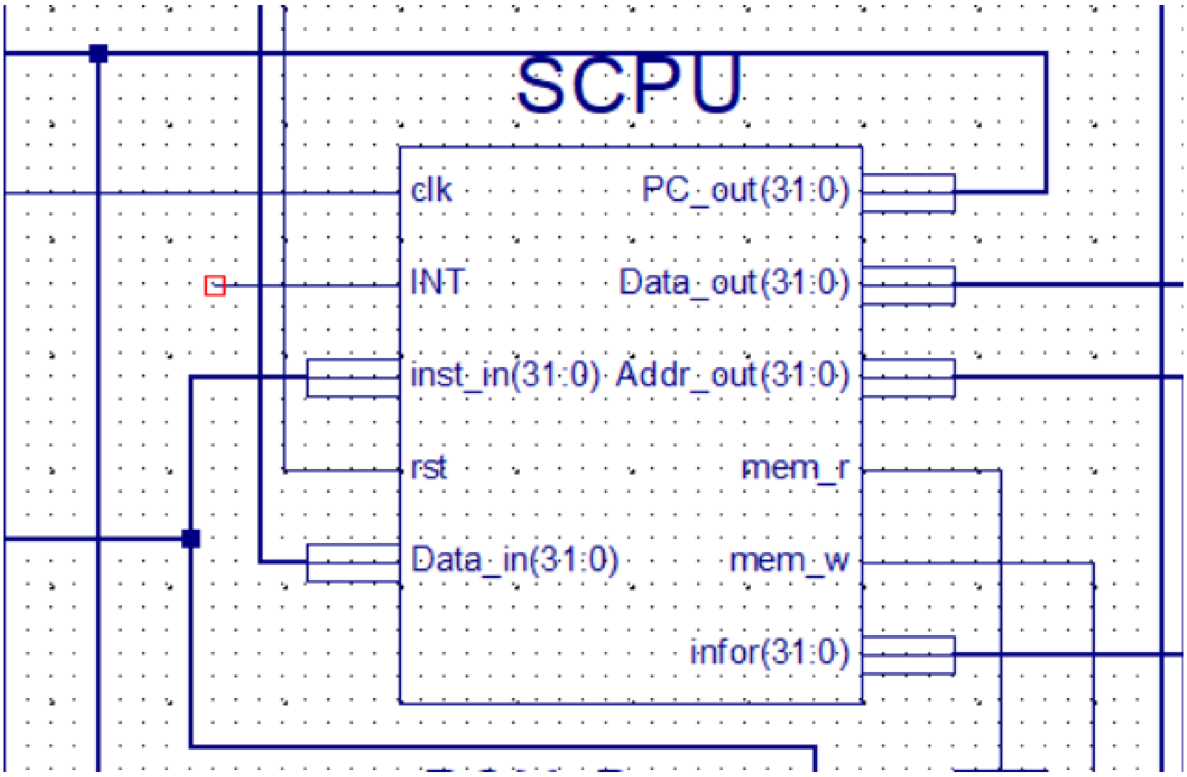
connected to PC register



change of ctrl and top

ctrl: add "assign eret=(opcode==6'b010000&&func==6'b011000)?1:0;"

top: connect INT to SW_OK[1]



mul

```
module mul(
          input clk,rst,
        input start,
        input [31:0] a,
        input [31:0] b,
        output done,
        output [31:0] yji,
          output [31:0] test
        );
reg[31:0] temp_c;
reg[31:0] temp_b;
reg[5:0] i;
reg done_r;

always @(posedge clk or posedge rst)begin
    if(rst) i = 6'd0;
    else if(start&&i<6'd33) i = i+1'b1;
    else i = 6'd0;
end

always @(posedge clk or posedge rst)
    if(rst) done_r = 1'b0;
    else if(i == 6'd32) done_r = 1'b1;
    else if(i == 6'd33) done_r = 1'b0;
assign done = done_r;

always @ (posedge clk or posedge rst)begin
    if(rst) begin
        temp_c = 64'h0;
        temp_b = 64'h0;
    end
    else if(start) begin
        if(i == 6'd0) begin
```

```
            temp_c = 0;
            temp_b = b;
        end
        else begin
            temp_c = temp_c << 1;
      if(temp_b[31]==1) temp_c = temp_c + a;
      else temp_c = temp_c;
            temp_b=temp_b<<1;
        end
     end
end

assign yji = temp_c[31:0];
assign test=temp_b;

endmodule
```

div

```
module div(
        input clk,rst,
        input start,
        input [31:0] a,
        input [31:0] b,
        output done,
        output [31:0] yshang,
        output [31:0] yyushu
        );
reg[63:0] temp_a;
reg[63:0] temp_b;
reg[5:0] i;
reg done_r;

always @(posedge clk or posedge rst)begin
    if(rst) i = 6'd0;
    else if(start&&i<6'd33) i = i+1'b1;
    else i = 6'd0;
end

always @(posedge clk or posedge rst)
    if(rst) done_r = 1'b0;
    else if(i == 6'd32) done_r = 1'b1;
    else if(i == 6'd33) done_r = 1'b0;
assign done = done_r;

always @ (posedge clk or posedge rst)begin
    if(rst) begin
        temp_a = 64'h0;
        temp_b = 64'h0;
    end
    else if(start) begin
        if(i == 6'd0) begin
            temp_a = {32'h00000000,a};
            temp_b = {b,32'h00000000};
        end
        else begin
            temp_a = temp_a << 1;
        if(temp_a >= temp_b) temp_a = temp_a - temp_b + 1'b1;
        else temp_a = temp_a;
          end
       end
end

assign yshang = temp_a[31:0];
assign yyushu = temp_a[63:32];
endmodule
```

change of datapath

```
    REG32  ALUREG (.CE(V5),
                  .clk(clk),
                  .D(mul?mul_out:res[31:0]),
                  .rst(NO),
                  .Q(ALU_out[31:0]));

     mul mull(.clk(clk),.rst(reset),
         .start(mul),
         .a(RdataA),
         .b(data_out),
         .done(mul_done),
         .yji(mul_out),
           .test());

     div divv(.clk(clk),.rst(reset),
         .start(div),
         .a(RdataA),
         .b(data_out),
         .done(div_done),
         .yshang(div_out1),
         .yyushu(div_out2));

     //assign test3=div_out1;
     //assign test2=div_out2;

     REG32  LO (.CE(Wdiv),
                  .clk(clk),
                  .D(div_out1),
                  .rst(NO),
                  .Q(test3));

     REG32  HI (.CE(Wdiv),
                  .clk(clk),
                  .D(div_out2),
                  .rst(NO),
                  .Q(test2));
```

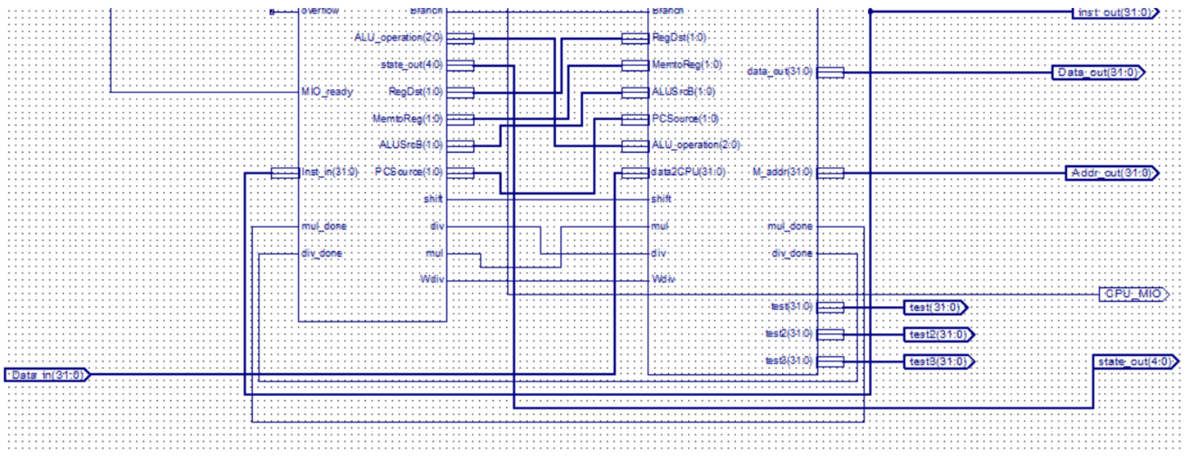
change of ctrl

```
     ID: begin case (Inst_in[31:26])
                 6'b000000: begin
                     case(Inst_in[5:0])
                             6'b000000: state<=EX_shift;
                             6'b000010: state<=EX_shift;
```

```
                            6'b011010: state<=EX_div; //div
                            6'b001000: state<=EX_jr; //jr
                            default:state <= EX_R; //R-type OP
                    endcase
                    end
                6'b011100: state <= EX_mul; //mul
                6'b000010: state <= Exe_J; // j
                6'b000011: state <= EX_JAL;// jal
                6'b000100: state <= EX_beq; // beq
                6'b000101: state <= EX_bne; // bne
                6'b001000: state <= EX_I; // addi
                6'b001010: state <= EX_I; // slti
                6'b001100: state <= EX_I;// andi
                6'b001101: state <= EX_I;// ori
                6'b001110: state <= EX_I;// xori
                6'b001111: state <= Lui_WB;// lui
                6'b100011: state <= EX_Mem; // lw
                6'b101011: state <= EX_Mem; // sw
                default:  state <= Error;
            endcase
            end
    EX_mul: begin if(mul_done) state<=WB_mul;
                   else state<=state;
                end
    EX_div: begin if ( div_done )state<=IF;
                        else state<=state;
                    end
    WB_mul: state<=IF;
```

```
    EX_div: begin ` CPU_ctrl_signals = 17'b00000000001000000; Branch=0; shift=0;
div=1; mul=0; Wdiv=1; end
    EX_mul: begin ` CPU_ctrl_signals = 17'b00000000001000000; Branch=0; shift=0;
div=0; mul=1; Wdiv=0; end
    WB_mul: begin ` CPU_ctrl_signals = value7; Branch=0; shift=0; div=0; mul=1;
Wdiv=0; end
```

change of top



四、 讨论与心得

In this experiment we are asked to realize mul/div and simple INT. With the convinience of verilog we can construct mul/div easily by coding instead of drawing schematic. And for INT, the main porpose is to realize eret instruction. I connect INT signal to SW_OK(1) so that we can stop the CPU using switch. And for the test coe, it will jump to a series of I-type instuctions and finally fall in to eret.

This is the end of experiments in this course. I'v been to the lab in school and debug all my projects. After finishing all the debugging, go back and see the reports before, I can really find many errors and finally careless mistakes. What's more, now I have a better understanding of top, datapath, ctrl and other modules instead of simply copy them from PPT. Then I realize the meaning of doing experiments offline.